Computing sparse permanents faster

Rocco A. Servedio Andrew Wan

Department of Computer Science, Columbia University, New York, NY 10027, USA

Abstract

Bax and Franklin (2002) gave a randomized algorithm for exactly computing the permanent of any $n \times n$ zero-one matrix in expected time $\exp\left[-\Omega\left(\frac{n^{1/3}}{2\ln n}\right)\right] 2^n$. Building on their work, we show that for any constant $C > 0$ there is a constant $\epsilon > 0$ such that the permanent of any $n \times n$ (real or complex) matrix with at most Cn nonzero entries can be computed in deterministic time $(2-\epsilon)^n$ and space $O(n)$. This improves on the $\Omega(2^n)$ runtime of Ryser's algorithm for computing the permanent of an arbitrary real or complex matrix.

Key words: algorithms, combinatorial problems, computational complexity

1 Introduction

1.1 Background

The permanent of an $n \times n$ real or complex valued matrix A is defined to be

$$\mathrm{per}(A) = \sum_{\sigma} a_{1\sigma(1)} \dots a_{n\sigma(n)}$$

where the sum ranges over all $n!$ permutations σ of $[n] = \{1, \dots, n\}$.

The permanent is an interesting and important characteristic of matrices. For example, the permanent of the adjacency matrix of a bipartite graph G is exactly the number of perfect matchings in G. Additionally, there are problems in statistical mechanics, quantum field theory, chemistry, combinatorics, and linear algebra that have been reduced to the computation of a permanent. Unfortunately, computing the permanent is believed to be quite hard, as Valiant proved in 1979 that even the problem when restricted to 0/1 matrices is #P complete [4]. On the other hand, a fully-polynomial randomized approximation scheme for computing the permanent of arbitrary matrices with non-negative entries was recently discovered by Jerrum, Sinclair, and Vigoda in [2].

Although we do not expect to find polynomial time algorithms for exactly computing the permanent, we can still hope to find algorithms with substantial improvements in efficiency over a naive approach, and progress in this direction has been made. Ryser's formula [3] gives the best known running time of $\Theta(n2^n)$ (instead of the trivial $n!$ algorithm) for arbitrary matrices. More recently in [1], Bax & Franklin gave a randomized algorithm for computing the permanent of 0/1 matrices that runs in expected time $\exp\left[-\Omega\left(\frac{n^{1/3}}{2\ln n}\right)\right]2^n$.

1.2 This work.

In this work we consider the problem of exactly computing the permanent of sparse matrices with real or complex entries. Even this problem must be hard; it is easily seen that computing the permanent of a 0/1 matrix with at most Cn ones is also #P complete. [1] We are not aware of previous work on algorithms that specifically target this problem. As our main result we give a deterministic algorithm B to compute the permanent of any sparse $n \times n$ matrix A, and prove the following theorem:

Theorem 1.1 *For any constant $C > 0$, there is a constant $\epsilon > 0$ such that algorithm B runs in at most $(2-\epsilon)^n$ time on any $n \times n$ matrix A with at most Cn nonzero entries.*

Theorem 1.1 improves on the $\Theta(n2^n)$ runtime of Ryser's algorithm, which as far as we know was the previous best algorithm for exactly computing the permanent of a sparse matrix.

1.3 Relation to previous work.

Our algorithm is an extension of Ryser's algorithm which exploits an idea in [1]. Bax and Franklin use a finite difference formula for the permanent (this is a generalization of Ryser's formula; like Ryser's formula, it expresses

[1] If we have an algorithm to compute the permanent of such a matrix that runs in time $f(n)$ for some function f, then we have an algorithm that computes the permanent for arbritrary 0/1 matrices that runs in time $f(n^2)$; the algorithm simply takes the arbitrary $n \times n$ matrix and embeds it in a diagonal $n^2 \times n^2$ matrix with the original matrix in the upper left corner instead of the ones. Since this matrix has the same permanent as the original matrix and at most $2n^2 - n$ ones, the original algorithm can be run on it to obtain the permanent in time $f(n^2)$.

the permanent as a sum over 2^n many terms), and show that by randomly augmenting the matrix in a particular way, the finite difference formula will have at most a $\exp[-\Omega(n^{1/3})]$ expected fraction of nonzero terms. They then show how to evaluate the formula while avoiding most of the zero terms.

Similar to the algorithm of Bax and Franklin, ours works by augmenting the matrix and then using Ryser's formula in a way that lets us avoid most of the zero-valued terms. The main differences between our work and [1] are: (1) We show that for sparse matrices, an appropriate augmented matrix can be chosen *deterministically* (i.e. we show how to efficiently derandomize a random augmentation of the matrix); (2) We show that our algorithm works for any (real or complex valued) sparse matrix, whereas their algorithm works for any 0/1 matrix; (3) We show that our algorithm requires only $O(n)$ space whereas their algorithm requires $2^{n/2}$ space.

2 Using Ryser's Formula

Ryser's well known formula gives an alternate expression for the permanent:

$$\operatorname{per}(A) = -1^n \sum_{S \subseteq [n]} (-1)^{|S|} \prod_{i=1}^{n} \sum_{j \in S} a_{ij}.$$

By ordering the subsets $S \subseteq [n]$ according to a Gray code, this yields a $\Theta(n \cdot 2^n)$ time algorithm for the permanent.

Our algorithm will compute $\operatorname{per}(A)$ using Ryser's formula on a different matrix A_b that has the same permanent as A. The time savings will come from creating A_b so that the number of nonzero terms in the outer sum will be $(2-\epsilon)^n$ instead of $\Theta(2^n)$. (Note that it is easy for a sparse matrix A to have $\Theta(2^n)$ many nonzero terms in Ryser's formula; for example, any 0/1 matrix with all 1s in the first column will have at least 2^{n-1} nonzero terms.) Since the term $\prod_{i=1}^{n} \sum_{j \in S} a_{ij}$ corresponding to a particular subset S of $[n]$ evaluates to zero whenever there is some i such that $\sum_{j \in S} a_{ij} = 0$, we construct A_b with the aim of increasing the fraction of terms which have some row summing to zero.

Given an $n \times n$ matrix A and an n-dimensional column vector b, as in [1] we let A_b be the $(n+1) \times (n+1)$ matrix

$$A_b = \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix}.$$

Note that A_b and A have the same permanent regardless of the choice of b. Using Ryser's formula, the permanent of A_b is

$$\mathrm{per}(A) = \mathrm{per}(A_b) = -1^{n+1} \sum_{S\subseteq[n+1]} (-1)^{|S|} \prod_{i=1}^{n+1} \sum_{j\in S} (a_b)_{ij}$$

Since the $(n+1)$-st row sum $\sum_{j\in S}(a_b)_{n+1,j}$ is either 1 or 0 depending on whether or not S contains $n+1$, we have that

$$\mathrm{per}(A) = \mathrm{per}(A_b) = -1^{n+1} \sum_{S\subseteq[n]} (-1)^{|S|} \prod_{i=1}^{n} \left(b_i + \sum_{j\in S} a_{ij} \right). \tag{1}$$

For any $S \subseteq [n]$ and $i \in [n]$, the i-th row sum for the term corresponding to S will be zero iff $\sum_{j\in S} a_{ij} = -b_i$. Therefore we will choose the vector b so that many terms have some row sum equal to zero.

2.1 Augmenting A to create many zero terms

We henceforth suppose that the matrix A has at most Cn nonzero entries. It is easy to see that at least half the rows of A must each have at most $2C$ nonzero entries. Since permuting rows does not change the value of the permanent, we may assume that these are rows $1, \dots, \frac{n}{2}$.

Now if there are at most $2C$ nonzero entries in row i of A, then over all 2^n choices of $S \subseteq [n]$ there are at most 2^{2C} distinct values that $\sum_{j\in S} a_{ij}$ can assume. Let $B_i = \{v : \sum_{j\in S} a_{ij} = -v \text{ for some } S \subseteq [n]\}$. Suppose we choose each coordinate b_i of b uniformly at random from B_i. Then for each $S \subseteq [n]$ and $i \in [\frac{n}{2}]$, the probability that $\sum_{j\in s} a_{ij} = -b_i$ is at least $\frac{1}{2^{2C}}$. Since each b_i was chosen independently, for each $S \subseteq [n]$ and any $m \leq \frac{n}{2}$ we have that $b_i + \sum_{j\in S} a_{ij} \neq 0$ for all $i = 1, \dots, m$ with probability at most $(1 - \frac{1}{2^{2C}})^m$. By linearity of expectation, we have that for a random choice of b as described above, the expected number of subsets $S \subseteq [n]$ for which $b_i + \sum_{j\in S} a_{ij} \neq 0$ for all $i = 1, \dots, m$ is at most $2^n \cdot (1 - \frac{1}{2^{2C}})^m$. Consequently there must be some particular choice of b for which at most $2^n \cdot (1 - \frac{1}{2^{2C}})^m$ many subsets have all nonzero row sums in the first m rows.

We can deterministically identify such a vector b by iterating over all $(|B_1| \cdot |B_2| \cdots |B_m|) \leq 2^{2Cm}$ many possibilities for $b_1, \dots, b_m$ (and setting $b_{m+1}, \dots, b_n$ arbitrarily). For each such b, we compute the number of subsets $S \subseteq [n]$ that have all nonzero row sums in the first m rows (and we ultimately take the b that minimizes this number). This is done as follows: let $T \subseteq [n]$ denote $\{j : a_{ij} \neq 0 \text{ for some } i \in [m]\}$, and note that $|T| \leq 2Cm$ since each of the first

m rows has at most $2C$ nonzero entries. For a given choice of b, the number of subsets $S \subseteq [n]$ that have all nonzero row sums in the first m rows equals $2^{n-|T|}$ times the number of subsets $S' \subseteq T$ that have all nonzero row sums in the first m rows. There are at most 2^{2Cm} such subsets S', and thus the total time required to choose the best b (based on the first m coordinates) is at most 2^{4Cm}. The procedure is easily seen to be space efficient, and thus we have the following:

Lemma 2.1 *Let A be any $n \times n$ matrix with at most Cn nonzero entries, and let $m \leq \frac{n}{2}$. The above deterministic procedure will construct an augmented matrix A_b such that the outer sum of (1) has at most $2^n \left(1 - \frac{1}{2^{2C}}\right)^m$ many S's such that $\sum_{j \in S} a_{ij} + b_i \neq 0$ for all $i = 1 \ldots m$. The procedure takes 2^{4Cm} time and $O(n)$ space.*

3 Avoiding Zero Value Terms

We now show how we can compute the permanent of A_b while avoiding many zero-valued terms. In selecting b we determined whether or not each $S' \subseteq T$ had all nonzero row sums in the first m rows. We will use the same information to determine which subsets $S \subseteq [n]$ to include in our sum. For a suitably chosen m, we will avoid enough zero-valued terms to achieve our running time.

Let $T \subseteq [n]$ be defined as in section (2). For each subset $S' \subseteq T$, check if $\sum_{j \in S'} a_{ij} + b_i = 0$ for some $i = 1, \ldots, m$. If some i yields 0, we move on to the next S' (since no subset $S \subseteq [n]$ which agrees with S' on T can contribute to (1)). Otherwise, if no i yields 0, then for each $R \subseteq [n] \setminus T$, let $S = S' \cup R$ and add to the running total

$$(-1)^{|S|} \prod_{i=1}^{n} \left(b_i + \sum_{j \in S} a_{ij} \right) \tag{2}$$

(i.e. add to the running total the term corresponding to every S for which membership in T is fixed according to S'). Finally, when we have gone through every $S' \subseteq T$, we multiply the total by $(-1)^{n+1}$ to obtain the permanent of A_b.

To bound the running time of this algorithm, we first note that we must check at most 2^{2Cm} subsets S' of T. Since any term (2) that we add to the running total must have nonzero row sums for each of the first m rows, by Lemma 2.1 there are at most $2^n \left(1 - \frac{1}{2^{2C}}\right)^m$ subsets $S = S' \cup R$ (across all $S' \subseteq T$) for which we compute (2). Taking $m = \frac{n}{8C}$, the runtime of this phase of the algorithm is

$$2^{n/4} + \text{poly}(n) \cdot 2^n (1 - \frac{1}{2^{2C}})^{n/(8C)} = 2^{n/4} + (2-\epsilon)^n \text{ for some } \epsilon > 0.$$

Note that we only need space to indicate the current S,S', compute the product of row sums, and the running total. Since constructing A_b takes time $2^{4Cm} = 2^{n/2}$ by Lemma 2.1, we have the following theorem:

Theorem 3.1 *For any $C > 0$, there exists $\epsilon > 0$ such that our algorithm can compute the permanent of any $n \times n$ matrix with at most Cn nonzero entries in time $O((2-\epsilon)^n)$ and space $O(n)$.*

4 Conclusion

We conclude by noting that our algorithm avoids the space requirement of $O(2^{n/2}n)$ in [1]. Since our algorithm is simple, deterministic, has a faster asymptotic runtime and uses no more space than a naive application of Ryser's formula, it may be a useful alternative to Ryser's formula for exactly computing the permanent of sparse matrices.

References

[1] E. Bax and J. Franklin. A permanent algorithm with $\exp[\omega(n^{1/3}/2 \ln n)]$ expected speedup for 0-1 matrices. *Algorithmica*, 32:157–162, 2002.

[2] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. In *Proceedings of the 33rd Annual Symposium on Theory of Computing*, pages 712–721, 2001.

[3] H.J. Ryser. *Combinatorial Mathematics*. Carus Mathematical Monograph No. 14. Wiley, 1963.

[4] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.